DOCUMENT RESUME

ED 345 712 IR 015 545

AUTHOR Henry, M. J.; Southerly, T. W.
TITLE A Comparison of the Language Features of Basic and HyperCard.
PUB DATE Mar 92
NOTE 15p.; Paper presented at the Annual Conference of the Eastern Educational Research Association (Hilton Head, SC, March 1992).
PUB TYPE Reports - Evaluative/Feasibility (142) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS Authoring Aids (Programing); Comparative Analysis; *Computer Assisted Instruction; Computer Software; *Hypermedia; Postsecondary Education; Problem Solving; *Programing; *Programing Languages
IDENTIFIERS *BASIC Programing Language; *HyperCard

ABSTRACT

This paper examines the structure of the Applesoft BASIC programming language and the Macintosh authoring language, HyperCard, and scrutinizes the language structures as the building blocks for moving along a chain of cognitive outcomes that culminates in the acquisition of problem solving skills which allow the programmer to learn new formal systems and transfer already learned skills to solving problems in other languages. The language features of each are summarized in terms of the concepts of mode, command statements, data storage, program control, math functions, and documentation, and similarities and differences are indicated. An instructional program written in HyperCard was duplicated as closely as possible in BASIC to use as a case study in order to examine the similarities and differences between the two languages. The content of the program consists of a telephone questionnaire from an applied social research study. Step-by-step instructions are given to the interviewer (the program user) as an interview is conducted. Getting the respondent on the phone, asking for permission to participate in the study, six survey questions, ways to document answers, and the interview's closing comments were included. It was found that, in general, although both languages allow for user friendly programs, overall HyperCard was easier to program to allow for user friendly features. (13 references) (DB)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION
CENTER (ERIC)

☐ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

ED345712

A Comparison of the Langauge Features of Basic and HyperCard

M. J. Henry and T. W. Southerly
West Virginia University

IR015545

Presented at
Eastern Educational Research Association Conference
Hilton Head, S. C. March 1992

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Mary Janet Henry

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

BEST COPY AVAILABLE

A Comparison of the Language Features of BASIC and HyperCard

M. J. Henry and T. W. Southerly
West Virginia University

The potential value of learning to program can have a dual nature. Besides learning to control the computer, the programmer is learning a form of problem-solving (Linn, 1985). The cognitive effects of programming instruction have been well documented (Linn, 1985; Mandinach & Linn, 1986; Dalbey & Linn, 1986). Generally speaking, programming has been described as having a series of stages, each requiring different cognitive activities. Specification, design, coding, and debugging are typical of the tasks involved in solving a programming problem (Dalbey & Linn, 1985).

Dalbey & Linn (1986) have provided a model of outcomes that could result from programming. Called a chain of cognitive outcomes, it indicates how various learning results are needed to produce technological expertise. Language features (instruction), design skills (access), and general problem solving skills (ability) are the three categories of potential learning outcomes included in the model. Instructional outcomes include knowledge of features and functions of the language being studied, i. e. specific, detailed statements and outcomes of those statements. Access outcomes are considered to be template and procedural skills. Templates are patterns or recurrent elements in a solution to similar problems. Procedural skills are used to combine templates and language features to invent and refine new programs. The chain of outcomes culminates in the acquisition of problem solving skills which allow the programmer to learn new formal systems and transfer already learned skills to solving problems in other languages.

Obviously the chain cannot be completed if the student does not have a good grasp of the features of the language. Putnam, Sleeman, Baxter, and Kispa (1986) have argued that a lack of knowledge of the basic features will prevent students from developing the higher-level cognitive skills. They continue by saying that learning enough about the

specific constructs of a language is a major component of learning to program. Sophisticated programming activities that require expertise require a good model of the structure of the language. (Kurland, Pea, Clement, & Mawby, 1986). In a study of programming recall, Magliaro and Burton (1988) found that beginning programmers tended to group like commands together syntactically, while advanced programmers attempted to recall programs in a procedural order.

A question posed by Dalbey and Linn (1986) was whether or not learners can be moved along the chain of cognitive consequences more efficiently and effectively. They investigated interventions to improve instruction by using methods that would assist the move from specific knowledge to general skills. Their implementation of a special purpose programming language, Spider World, allowed students to plan and test programs without spending too much time learning language features. Results showed that students, after 12 weeks of treatment, benefited in different ways from their respective instruction, but that beyond limited gains in language features, none moved far along the chain of cognitive accomplishments.

In describing computer-assisted instruction, Cook (1989-90) has asserted that media in itself does not influence learning—he feels that content and method do, however. Content is usually an important component of any learning activity. The purpose of this paper is to examine the structure of the Applesoft BASIC programming language and the Macintosh authoring language, HyperCard. The language features of each will be presented and then the same program written in each language will be examined for comparison purposes. In effect, language structures will be scrutinized as the building blocks for moving along the chain of cognitive outcomes.

BASIC

BASIC was first developed at Dartmouth in the middle 1960's. Its widespread use in schools has been the reason researchers have often used this as the vehicle for their programming studies (Dalbey & Linn 1986). Critics of BASIC have suggested that many

of its features are not easily used in designing well-organized programs (Linn, 1985). However, efficient program design can be accomplished though a top-down instructional approach using modular program construction, according to research in this area (Kurland,Clement, Mawby & Pea, 1986; Magliaro & Burton, 1988; Reed, Palumbo & Stolar, 1988).

BASIC has two modes: (a) immediate—used for results that are desired as soon as the command is issued and (b) delayed—used for sequencing program lines and delaying action of the commands until the program is completed and then run. Programs can be constructed in text or graphic format. Two forms of graphics in BASIC are High-Resolution and Low-Resolution. The distinction between the two is the number of pixels (dots) on the screen which are involved. Both can present animation with the use of appropriate command statements.

Data storage is accomplished with the use of numeric or alpha-numeric (string) variables and arrays. Assignment, INPUT, and READ/DATA statements are the language features that allow the data to be deposited in computer memory. Mathematical manipulation of data is done with the usual math functions of addition, subtraction, division, and multiplication which can be used in a number of different ways to achieve desired results. The LEN command will create a centering routine for text; the RND command will randomize numbers; and the INT command will round off decimal numbers.

Program control statements allow the programmer to govern the flow of action with the use of loops (FOR...NEXT), conditionals (IF...THEN and GOTO xx), and subroutines (GOSUB). The logical operators AND and OR are used in conjunction with conditional statements for program flow. Internal documentation, which allows clarity upon later examination of the program, is performed by the use of REM statements.

Command statements include PRINT and its accompanying screen formatting commands of VTAB, HTAB, FLASH, INVERSE, SPEED = XXX, NORMAL, and END. PRINT allows specified output to the screen; VTAB changes the vertical position

on the screen, while HTAB changes the horizontal position; FLASH rapidly alternates background and foreground colors; INVERSE changes the background; SPEED=XXX allows the programmer to control the speed with which text is presented on the screen (XXX being any number up to 255); NORMAL negates the INVERSE and FLASH commands; and END is the last statement of a program. Accompanying DOS (disk operating system) commands such as LIST, RUN, NEW, HOME, and SAVE must also be included in the acquisition of the language structure of BASIC. LIST gives a line by line listing of the current program in RAM ("temporary" memory of the computer); RUN will allow the program to be performed; HOME will clear the screen; and NEW will clear RAM.

Graphics command statements include GR for low resolution and HGR for high resolution graphics which allow entry to the graphic screen and mode. COLOR= XXX (low-resolution) and HCOLOR=XXX (high resolution) are the commands for changing background and drawing colors. In low resolution the programmer is given a choice of 16 colors while in high resolution graphics, the choice is limited to six. Graphics programs can be linked to text programs with the use of embedded DOS (disk operating system) commands.

In summarizing the language features of Applesoft BASIC, the concepts of mode,command statements, data storage, program control, math functions, and documentation have been addressed. Next a relatively new authoring language, HyperCard, will be summarized using the same concepts.

HyperCard

HyperCard is the most widely used Macintosh authoring language and can be described as a "hypertext generator based on the concepts of hypermedia functioning as an intuitive personalized database" (Cook, 1989-90, pp. 112-113). Basically, it is "a software product that serves as a platform for constructing computer applications that integrate text, numbers, sound, and graphics within an information storage and retrieval system" (Adams,1990, p. 22). Harris has called HyperCard a "software erector set" (1988, p.33),

one that can emulate the way a person thinks . He feels that teachers who have no programming experience can customize their information to fit an individual lesson and that HyperCard is capable of incorporating content acquisition with active involvement of the student in the thinking process. La Passardiere (1989) describes it as a developmental tool and a database connected to an object-oriented language. Harris and Cady (1988) state that the set of pre-programmed linking tools present in HyperCard accounts for its ease of authoring. New materials can be added to the source material (content) in a HyperCard database and new links and paths can be added so that the user can browse through it in any direction desired. Thus, one can rapidly pursure a wide variety of topics and subtopics (Crane & Mylan, 1988).

The Macintosh environment allows HyperCard certain functions such as mouse-driven commands, icons, and pull-down menus. The objects in HyperCard which act as building blocks for structuring programs are *cards*—individual screens holding textual or graphical information or data (record); *stacks*—cards of related information (file); *background*—domain of all cards that defines its format and appearance; *fields*—contains text of either alphabetic or numeric data; *buttons*—link among cards and stacks; an area of the screen that is sensitive to a pointing device (mouse) and transfers control to another card. Stacks can be linked and control passed between them with the use of the Home Card which serves as an index and visually represents all stacks that can be accessed within the HyperCard system (See Figure 1).

The programming language of HyperCard is HyperTalk. Scripts are used to control stack processing and allow you to control the actions of HyperCard objects. Links are established by keywords embedded within displayed text and then linking these words with screens containing further information on the topic. Whenever an object is activated, a card is opened and the script for that object is executed, causing the scripted action to take place. Both text and graphics modes can be accommodated on the same card in a stack. Graphics can be imported from another application, another file, or drawn by the

programmer using the Tool menu. Animation can be accomplished by three methods which include: (a) showing all cards, (b) a repeat loop, or (c) selecting an object and scripting it to be "dragged" from one set of coordinates to another.

A number of command statements control various actions in the HyperCard stack. Commands that allow the user to navigate through the stack include Go, Find, Push and Pop. The Go command can be used to script a button to take the user to any card in the stack; the Find command will retrieve a text string or a variable; Push and Pop commands can be used within scripts to control the sequencing of card viewing and create modularity for the program. On, Set, Show and Hide are object manipulation commands through which the programmer can control the visual display on the screen. On is the structure that receives and takes action on messages while Set is the command that permits user levels to be changed. User levels include browsing, typing, painting, authoring, and scripting each allowing a higher level of programming control. The Hide command allows any object to be hidden from view on the screen; its opposite command, Show, makes visible an object which has been hidden.

Screen manipulation commands include Visual, Ask, and Answer. Visual allows varied optical effects in transition from one card to the next. The Ask and Answer commands are methods of soliciting information from the user. Dialog boxes appear containing a prompt asking for specified information with the Ask command; the Answer command can supply up to three responses to a prompt.

Action commands include Wait, Put, Drag, Sort, DoMenu, and Choose. Wait causes any action to be delayed for a specified time; Put is one way to assign values to variables and to fields; Drag is an animation command to move from one spot to another; Sort arranges cards; DoMenu permits you to write scripts containing any of the menu commands; and Choose allows the choice of various tools from the tool menu.

Local and global variables are used for data storage; the PUT command is the accompanying statement that assigns the value to the variable. Arithmetic operators include

addition, subtraction, multiplication, division with the usual remainder, division producing a whole number, mod division ignoring the whole number and producing the remainder, exponential operator, and parentheses used to group expression to control order of evaluation. Logical operators include Is, Contains, Is in,True, False, and Empty each of which return certain values to be used by the program. Combining logical operators with the operators And, Or, and Not, along with parentheses allow the programmer to construct complex expressions.

Program control loops include Repeat; Repeat ..Until; Repeat...While; and Repeat ...With. Conditional statements which contribute to program control are If...Then and If...Then...Else. Internal documentation is the inclusion of two dashes (--) in front of any statement. Figure 2 presents the language features of BASIC and HyperCard.

To illustrate how both languages address a similar problem, a simple animation program in low-resolution graphics BASIC would involve the use of the FOR...NEXT loop and a variable. The same program in HyperCard could be scripted in a button and include the Drag command. The BASIC program lines to accomplish the action of moving a box on the screen from left to right include:

```
10 GR  [Sets graphics mode]
20 FOR BOX = 0 to 35  [Box is drawn at 36 different points across the screen]
30 COLOR=2  [Sets color for box]
40 GOSUB 1000  [Subroutine to draw box]
50 COLOR=0  [Sets color to default background color ]
60 GOSUB 1000  [Because the box is now being drawn in the background
                color, it will be "erased", thus seeming to move]
70 NEXT BOX  [Sends control back to line 20 until 36 iterations are completed]
999 END
1000 REM **********BOX**********
```

```
1010 HLIN BOX,BOX+4 AT 7  [Use of variable BOX to represent points from 0
1020 HLIN BOX,BOX +4 AT 8       to 35 in loop; changes the x value]
1030 HLIN BOX,BOX +4 AT 9
1040 RETURN
```

Programming in HyperCard involves drawing the box first, then dragging it to the desired location by scripting a button. The steps include:

1. Choose rectangle tool from Tools Menu. Draw box in desired area of screen.
2. Choose button tool from Objects Menu. Create button and script the following:

```
on mouseup
        Choose select tool
        Drag from 10,80 to 100,200  [selects area on screen to be moved]
        Set dragspeed to 200  [sets speed for animation]
        Drag from 10,200 to 400,200  [x,y values for old and new position]
        Choose browse tool  [returns control to browsing mode]
end mouseup
```

Similarities and Differences

HyperCard is mouse-driven in the Macintosh environment while on the Apple, the commands must be typed into the computer's memory. Both HyperCard and BASIC have text and graphics modes and both can accommodate animation in graphics. The speed of animation is controlled by a STEP command in BASIC while in HyperCard the command Set Dragspeed is used. The Home Card in HyperCard is comparable to the CATALOG listing in BASIC and it serves to link programs (stacks) as does the embedded DOS command in BASIC. Data is stored in arrays in BASIC while in HyperCard it is stored in stacks. Program modularity for top-down programming is accomplished in BASIC through the use of subroutines and the GOSUB statement; in HyperCard the comparable modularity is the Push and Pop commands.

High school honors students from a mid-Atlantic state received HyperCard training at a summer Honors Academy and were asked to enumerate any similarities and/or differences between HyperCard and BASIC (Reed, 1991). Their responses regarding ease of use contained expressions such as "similar in linking processes, but HyperCard is much easier to work with"; "creation of programs is much easier in HyperCard"; "HyperCard is much simpler and faster to use and learn"; and "application is user friendly."

In relation to actual programming, comments were found such as: "HyperCard doesn't appear to require as much technological knowledge about programming. It doesn't rely on symbols as much"; "mouse allows you to do with a button click what BASIC would require lines of code for"; "icons make tasks easier than typing in commands"; "the ability to move through the program while it is being designed makes the programming process less complicated"; "placing and moving text, graphics, or whole cards is simpler than trying to pick out individual parts of the BASIC program"; "HyperCard can use pictures not drawn in the programming."and "HyperCard is able to create pictures and backgrounds with its own programming before the program is run." [1]

Case Study

An instructional program written in HyperCard was duplicated as closely as possible in BASIC to use as a case study in order to examine the similarities and differences between the two languages. The program was originally designed in HyperCard to illustrate to the user how the computer may be useful in the process of conducting a social research telephone survey. Content of the program consists of a telephone questionnaire from an applied social research study. Step-by-step instructions are given to the interviewer (the program user) as an interview is conducted. Getting the respondent on the phone, asking for permission to participate in the study, six survey

---

[1] Quotes are from transcriptions of assignments.

questions, ways to document answers, and the interview's closing comments were included.

In many cases, HyperCard entails orchestrating pre-existing programming to suit individual purposes while the BASIC language requires the programmer to create and direct simultaneously. For example, in HyperCard the introductory text n the case study was typed with the aid of preprogrammed tools. Menus and options such as centering text; different text styles and size; and cut, copy, and paste text were selected by a simple click of the mouse. BASIC required that all features needed for text entry and manipulation be written in lines of command codes. Although HyperCard allows the user to combine language features to form a program that performs some activity or solves a problem, many of HyperCard's functions are pre existing. BASIC allows many similar activities or problem solving strategies, but the programmer must write code for all of the required functions.

In general, both languages allowed for user friendly programs. In both, it is possible to allow the user to move around within the programs with the use of conditionals and optional directions. For example, it was possible to allow the user to move from the menu to various parts of the survey and back in both programs. But, overall, HyperCard was easier to program than BASIC to allow for user friendly features. The use of fields in HyperCard was convenient and easy to program for features such as changing the respondents' name and number for multiple interviewing. Field use in HyperCard was also a great facilitator in allowing the user to input answers form the multiple choice responses.

One of the major differences between the programs was found in the telephone process of the survey. In HyperCard, there is a special phone feature which allows for automatic dialing. A graphic representation of a telephone was sensitized through the use of fields to list the name, identification number, and telephone number of the person called. A click on the telephone visual allowed the user to call automatically through the computer;

in BASIC, the automatic telephone feature was not possible. Instead, the monitor was utilized to list the respondents' name and number, and to direct the interviewer to manually dial the telephone.

Another major difference found between HyperCard and BASIC is in the use of graphics. In the HyperCard program, it was possible to copy a well-drawn picture from the special feature, Art Ideas. In the BASIC program, the picture was omitted. It would have been possible to utilize a picture by drawing one with the BASIC graphic features, but the programmer would need good drawing skills in order to obtain a quality picture. A similar difference was also found in the use of Button icons as graphic descriptors in HyperCard which were not available in BASIC.

Conclusion

In order to acquire design skills and problem-solving skills one must first be fluent in the language itself. Without basic knowledge of the command statements, program control, and modularity, the programmer will probably have a difficult time advancing along the cognitive course of outcomes described earlier.

The language features of BASIC and HyperCard have been presented and then contrasted for similarities and differences. Whether one is preferable to the other probably depends upon the purpose of learning the language. One must ask what the objective is for the study of programming.

References

Adams, D. R. (1990). HyperCard and Macintosh: A primer. New York: McGraw-Hill.

Cook, E. K. (1989-90). The use of Macintosh authoring languages in effective computer-assisted instruction. *Journal of Educational Technology Systems, 18* (2), 109-122.

Dalbey, J. & Linn, M. C. (1985). The demands and requirements of computer programming: A literature review. *Journal of Educational Computing Research , 1* (3), 253-274.

Dalbey, J. & Linn, M. C. (1986). Cognitive consequences of programming: Augmentations to BASIC instruction. *Journal of Educational Computing Research, 2* (1), 75-93.

Harris, M. & Cady, M. (1988). The dynamic process of creating hypertext literature. *Educational Technology*, ???

Kurland, D. M., Pea, R. D., Clement, C., & Mawby, R. (1986). A study of the development of programming ability and thinking skills in high school students. *Journal of Educational Computing Research, 2* (4), 429-458.

La Passardiere, B. (1989). *Lecture notes on computer science: Computer assisted learning*. Springler Verlag: Gt. Goos & J. Hartmanis.

Linn, M. C. (1985). The cognitive consequences of programming instruction in classrooms. *Educational Researcher, 14* (5),14-25-29.

Magliaro, S. B., & Burton, J. K. (1988). Adolescents' chunking of computer programs. *Computers in the Schools, 4* (3/4), 129-138.

Mandinach, E. B. & Linn, M. C. (1986). The cognitive effects of computer learning environments. *Journal of Educational Computing research, 2* (4), 411-427.

Putnam, R. T., Sleeman, D., Baxter, J. A., & Kuspa, L. K. (1986). A summary of misconceptions of high school BASIC programmers. *Journal of Educational Computing Research, 2* (4), 459-471.

Reed, W. M. (1991, June-July). Governor's Honors Academy. Morgantown: West Virginia University.

Reed, W. M., Palumbo, D. B. & Stolar, A. A. (1988). The comparative effects of BASIC and Logo instruction on problem-solving skills. *Computers in the Schools*, *4*(3/4), 105-118.